



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE

United States Patent and Trademark Office

Address: COMMISSIONER FOR PATENTS

P.O. Box 1450

Alexandria, Virginia 22313-1450

www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/730,975	12/10/2003	Efstratios Tsantilis	11884/406-401	5120
53000	7590	01/06/2009		
KENYON & KENYON LLP 1500 K STREET N.W. WASHINGTON, DC 20005				
EXAMINER				
WANG, JUE S				
ART UNIT		PAPER NUMBER		
2193				
MAIL DATE		DELIVERY MODE		
01/06/2009		PAPER		

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

**Office Action Summary****Application No.**

10/730,975

**Applicant(s)**

TSANTILIS, EFSTRATIOS

**Examiner**

JUE S. WANG

**Art Unit**

2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --  
**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 20 October 2008.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-35 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-35 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
  2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-946)
- 3) ☐ Information Disclosure Statement(s) (PTO/SF/ICE)  
Paper No(s)/Mail Date \_\_\_\_\_
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date \_\_\_\_\_
- 5) ☐ Notice of Informal Patent Application
- 6) ☐ Other: \_\_\_\_\_

**DETAILED ACTION**

1. Claims 1-35 have been examined.

***Claim Rejections - 35 USC § 103***

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

1. Claims 1-11, 33, and 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Schwabe (US 6,986,132 B1), in view of Atallah et al. (US 7,069,474 B2, hereinafter Atallah), further in view of Lin et al., "Multiuser Collaborative Work in Virtual Environment based CASE Tool" (hereinafter Lin), further in view of Boonsiri et al., "Automated Component Ensemble Evaluation" (hereinafter Boonsiri).

2. As per claim 1, Schwabe teaches a method for monitoring updates to a software repository, comprising:

downloading interfaces to a stored software component, the interface having respective versions of a software interface over a network from a software repository via an application programming interface (see Fig 11A, 11C, Fig 20C, column 17, lines 36-67, column 19, lines 19-27, column 24, lines 24-32, column 25, lines 23-36) ;

comparing a first downloaded version and a second downloaded version of the interfaces to detect at least one difference between them (see Figs 20C-20D, column 24, line 35-column 26, line 15); and

issuing an alert message containing an overall backward compatibility (see column 25, line 23 – column 26, line 15).

Schwabe does not explicitly teach snapshots of the software interface, and comparing the snapshots. However, a snapshot is considered as a copy of the software interface made at a particular time and applications for taking snapshots of a file are well known in the field, so it would have been obvious to one of ordinary skill in the art that a snapshot of the API definition file can be easily assembled and that comparing two snapshots of two software interfaces would be equivalent to comparing the two software interfaces directly.

Schwabe does not teach rating each detected difference according to a backward compatibility metric, determining an overall backward compatibility of the updated version based on the difference ratings.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including rating each detected difference according to a compatibility metric and determining an overall compatibility of the software products based on the difference ratings (see Figs 4A, 4B, abstract, lines 2-7, 11-14, column 5, line 64 - column 7, line 15).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to rate each detected difference according to a backward compatibility metric and determine an overall backward compatibility of the updated version based on the difference ratings as taught by Atallah because it is expensive and time consuming

to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow users to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

Schwabe and Atallah do not teach issuing an alert message to registered authors of the software component in a multi-author software design environment.

Lin teaches issuing alert messages to registered authors of software components in a multi-author software design environment to indicate conflicts (see page 259, right column, paragraph 1, page 264, right column, page 265, left column).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe and Atallah to perform the compatibility check in a multi-author environment such that the alert messages are issued to registered authors of the software component as taught by Lin because it is well known in the art that it is desirable to check version compatibility during software development.

Schwabe, Atallah, and Lin do not explicitly teach the use of a score to indicate the overall backward compatibility.

Boonsiri teaches a design environment that provides compatibility scores for different ensembles of software components (see page 40, section 1, paragraph 1, pages 41-42, section 2, page 42, section 3, last paragraph, pages 43-44, sections 3.1, 3.2, pages 50-51, section 6).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe, Atallah, and Lin to provide a compatibility score to indicate the overall backward compatibility as suggested by Boonsiri because the score can be used to provide a ranking (see page 51, paragraph 3 of Boonsiri).

3. As per claim 2, Schwabe teaches that the alert message is issued only when the overall backward compatibility indicates the updated version is not backward compatible (see column 25, line 23 – column 26, line 15).

4. As per claim 3, Schwabe does not teach that the compatibility metric comprises a table of software modifications including backward-compatible software modifications and backward-incompatible modifications.

Atallah teaches maintaining tables used during the risk assessment process to determine the compatibility of binary files (see column 6, lines 22-28, 34-39, 46-50). In determining backward compatibility between API definition files, Schwabe must have knowledge of what modifications preserve backward compatibility and what modifications break backward compatibility. It would have been obvious to one of ordinary skill in the art to maintain the backward compatible and backward incompatible modifications used in Schwabe in a database or table as taught by Atallah to facilitate easy access and maintenance.

5. As per claim 4, Schwabe further teaches that the backward incompatible software modifications include: deleting a parameter from a subroutine (see Fig 20D, column 10, lines 48-

50 and column 24, lines 1-9); and deleting a field from a public data structure (see Fig 20D, column 25, lines 49-58).

6. As per claim 5, Schwabe further teaches that the backward incompatible software modifications include: adding a mandatory parameter from a subroutine (see Fig 20D, column 10, lines 48-50 and column 24, lines 1-9; EN: adding a mandatory parameter to the method changes the parameters of the method); and adding a mandatory field to a public data structure (see Fig 20D, column 25, lines 49-58).

7. As per claim 6, Schwabe further teaches that the backward-incompatible software modifications include: redefining an optional parameter as a mandatory parameter(see Fig 20D, column 10, lines 48-50 and column 24, lines 1-9); changing a parameter data type (see Fig 20D, column 26, lines 6-10); and changing a public field data type (see Fig 20D, column 26, lines 1-10).

8. As per claim 7, this claim contains limitations that are substantially similar to claim 1. Therefore, it is rejected using the same reasons as claim 1.

9. As per claim 8, Schwabe does not teach that the alert message includes a summary of each detected difference.

Atallah teaches that the alert message includes a summary of each detected difference (see Fig 4A, Fig 4B, column 6, line 31 - column 7, line 16).

10. As per claim 9, Schwabe further teaches that the detecting comprises discovering that a parameter in a version is missing from a successive version (see Fig 20D, column 10, lines 48-50 and column 26, lines 6-10; EN: changing the parameters to a method includes deleting one of the parameters).

11. As per claim 10, Schwabe further teaches that the detecting comprises discovering that a parameter is optional in a version, but the parameter is mandatory in a successive version (see Fig 20D, column 10, lines 48-50 and column 26, lines 6-10; EN: changing the parameters to a method includes changing a parameter from optional to mandatory).

12. As per claim 11, Schwabe further teaches that the detecting comprises discovering that a parameter in a version is defined as a different data type in a successive version (see Fig 20D, column 10, lines 48-50 and column 26, lines 6-10).

13. As per claim 33, Schwabe teaches the API definition file comprises: the relevant data objects, subroutines, and associated attributes; where the proceeding are compiled for a selection of interfaces as well as any interfaces that are associated with the selected interfaces (see Fig 20D, steps 1635- 1655, column 25, line 37 - column 26, line 10; EN: the selected interfaces are the implemented interfaces and the interfaces associated with the selected interfaces are the superinterfaces). While Schwabe does not explicitly teach a snapshot of the API definition file, it would have been obvious that a snapshot of the API definition file would have the same



information as the API definition file because the snapshot is considered as just a copy of the API definition file.

14. As per claim 35, Schwabe does not teach wherein the alert message is issued if no difference is detected between the two snapshots being compared.

Atallah teaches issuing an alert message if no difference is detected between the two snapshots being compared (see column 6, line 64 – column 7, line 16).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to issue an alert message if no difference is detected between the two snapshots being compared as taught by Atallah because the report enables the user to better gauge binary compatibility issues (see column 7, lines 2-15 of Atallah).

15. Claims 12 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Schwabe (US 6,986,132 B1), in view of Lin et al., “Multiuser Collaborative Work in Virtual Environment based CASE Tool” (hereinafter Lin).

16. As per claims 12, Schwabe teaches the invention as claimed, including a method for monitoring updates to a software repository, comprising:

downloading snapshots of a software object where each snapshot is of a respective version (see Fig 11A, 11C, Fig 20C, column 17, lines 36-67, column 19, lines 19-27, column 24, lines 24-32, column 25, lines 23-36; EN: the API are considered as a snapshot of the actual object) ;

comparing a first downloaded version and a second downloaded version of the snapshots(see Figs 20C-20D, column 24, line 35-column 26, line 15); and

detecting at least one difference between the first downloaded version of an object interface and the second downloaded version of the object interface (see Figs 20C-20D, column 24, line 35-column 26, line 15);

issuing an alert message when at least one of the detected differences indicates the successive version is not backward compatible (see column 25, line 23 – column 26, line 15).

Schwabe does not teach issuing an alert message to registered authors of the software component in a multi-author software design environment.

Lin teaches issuing alert messages to registered authors of software components in a multi-author software design environment to indicate conflicts (see page 259, right column, paragraph 1, page 264, right column, page 265, left column).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to perform the compatibility check in a multi-author environment such that the alert messages are issued to registered authors of the software component as taught by Lin because it is well known in the art that it is desirable to check version compatibility during software development.

17. As per claim 13, Schwabe further teaches that the object interface comprises a list of components published by a software object residing in the object repository, said components including object properties and object methods (see Figs 11A, 18, Fig 20D, column 11, lines 51-57, and column 17, line 48 – column 18, line 5).

18. Claims 14-28 and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over Schwabe (US 6,986,132 B1), in view of Atallah et al. (US 7,069,474 B2, hereinafter Atallah), further in view of Lin et al., "Multiuser Collaborative Work in Virtual Environment based CASE Tool" (hereinafter Lin).

19. As per claim 14, Schwabe teaches the invention as claimed, including a method for monitoring updates to software interfaces, comprising:

detecting at least one difference between a first software interface of a stored software component and a second software interface of a second stored software component (see Figs 11A, 20C-20D, column 17, lines 36-67, column 24, line 35-column 26, line 15);

issuing an alert when at least one of the detected differences indicates the first software interface is not backward compatible with respect to the second software interface (see column 25, line 23 – column 26, line 15).

Schwabe does not explicitly teach taking snapshots of the software interface, and comparing the snapshots. However, a snapshot is considered as a copy of the software interface made at a particular time and applications for taking snapshots of a file are well known in the field, so it would have been obvious to one of ordinary skill in the art that a snapshot of the API definition file can be easily assembled and that comparing two snapshots of two software interfaces would be equivalent to comparing the two software interfaces directly.

Schwabe does not teach rating each detected difference according to a backward compatibility metric.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including rating each detected difference according to a compatibility metric and determining an overall compatibility of the software products based on the difference ratings (see Figs 4A, 4B, abstract, lines 2-7, 11-14, column 5, line 64 - column 7, line 15).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to rate each detected difference according to a backward compatibility metric as taught by Atallah because it is expensive and time consuming to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow users to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

Schwabe and Atallah do not teach issuing an alert message to registered authors of the software component in a multi-author software design environment.

Lin teaches issuing alert messages to registered authors of software components in a multi-author software design environment to indicate conflicts (see page 259, right column, paragraph 1, page 264, right column, page 265, left column).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe and Atallah to perform the compatibility check in a multi-author environment such that the alert messages are issued to registered authors of the software component as taught by Lin because it is well known in the art that it is desirable to check version compatibility during software development.

20. As per claim 15, Schwabe does not teach rating each detected difference according to a predetermined difference metric; determining an overall difference between the first software interface and the second software interface based on the difference ratings; and incorporating the overall difference into the alert message.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including rating each detected difference according to a compatibility metric and determining an overall compatibility of the software products based on the difference ratings (see Figs 4A, 4B, abstract, lines 2-7 and lines 11-14, column 5, line 64 - column 7, line 15).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to rate each detected difference according to a predetermined metric and determine an overall difference between the first interface and the second interface based on the difference ratings as taught by Atallah because it is expensive and time consuming to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow users to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

21. As per claim 16, Schwabe does not teach assigning a user to the first snapshot; and issuing the alert message to the user.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including assigning a user to the application being assessed (see column 5, lines 40-63) and issuing the risk assessment report to the user (see column 7, lines 5-16).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to assign a user to the first snapshot; and issuing the alert message to the user as taught by Atallah because it is expensive and time consuming to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow users to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

22. As per claim 17, Schwabe does not teach incorporating a summary of each detected difference into the alert message.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including incorporating a summary of each detected compatibility difference into a report (see Fig 4A, Fig 4B, column 6, line 31 - column 7, line 16).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to incorporate a summary of each detected difference into the alert message as taught by Atallah because it is expensive and time consuming to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new

version of the operating system, so the risk profile generated will allow user to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

23. As per claim 18, Schwabe further teaches that the API definition file has at least one record, said record including at least one attribute (see Fig 20D, column 25, line 37 – column 26, line 10; EN: attributes of classes, interfaces, fields and methods, are all considered as separate records in the API definition file) While Schwabe does not teach organizing the records in a table, it would have been obvious that a table structure can be used since a table is a well known data structure in the art. Furthermore, it would have been obvious that a snapshot of the API definition file would have the records because the snapshot is considered as just a copy of the API definition file.

24. As per claim 19, Schwabe further teaches that the record describes an object property (see Fig 20D, column 25, lines 49-58; EN: the fields of a class are considered as properties of the class since the fields of a class describe the class and class properties are considered object properties because it is well known in the art that an object is just an instance of a class).

25. As per claim 20, Schwabe further teaches that the attribute indicates a data type of the object property (see Fig 20D, column 25, lines 49-58, column 26, lines 1-5; EN: the data type of the fields are considered data type of the object property).

26. As per claim 21, Schwabe further teaches that the attribute indicates a data length of the object property (see Fig 20D, column 25, lines 49-58, column 26, lines 1-5; EN: length information is implied by the type information since a type of int requires a different amount of storage than a type of float).

27. As per claim 22, Schwabe further teaches that the record describes an object method (see Fig 20D, item 1655, and column 26, lines 6-10).

28. As per claim 23, Schwabe further teaches that the records describe an object method parameter (see Fig 20D, item 1655, and column 26, lines 6-10; EN: the object method parameter is described in the method signature).

29. As per claim 24, Schwabe further teaches that the attribute indicates a data type of the object method parameter (see Fig 20D, item 1655, and column 26, lines 6-10; EN: the data type of the object method parameter is described in the method signature).

30. As per claim 25, Schwabe teaches a computer programmed to monitor versions of a software interface, comprising:

means to download interfaces to a stored software component, the interface having respective versions (see Fig 11A, 11C, Fig 20C, column 17, lines 36-67, column 19, lines 19-27, column 24, lines 24-32, column 25, lines 23-36) ;



means to compare a first downloaded version and a second downloaded version of the interfaces (see Figs 20C-20D, column 24, line 35-column 26, line 15);

means to detect at least one difference between the interfaces (see Figs 20C-20D, column 24, line 35-column 26, line 15); and

means to issue an alert message containing an overall backward compatibility (see column 25, line 23 – column 26, line 15).

Schwabe does not explicitly teach snapshots of the software interface, and comparing the snapshots. However, a snapshot is considered as a copy of the software interface made at a particular time and applications for taking snapshots of a file are well known in the field, so it would have been obvious to one of ordinary skill in the art that a snapshot of the API definition file can be easily assembled and that comparing two snapshots of two software interfaces would be equivalent to comparing the two software interfaces directly.

Schwabe does not teach means determine an overall backward compatibility of the second downloaded version of the interface based on the difference ratings.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including rating each detected difference according to a compatibility metric and determining an overall compatibility of the software products based on the difference ratings (see Figs 4A, 4B, abstract, lines 2-7, 11-14, column 5, line 64 - column 7, line 15).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to rate each detected difference according to a backward compatibility metric and determine an overall backward compatibility of the updated version based on the difference ratings as taught by Atallah because it is expensive and time consuming

to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow users to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

Schwabe and Atallah do not teach issuing an alert message to registered authors of the software component in a multi-author software design environment.

Lin teaches issuing alert messages to registered authors of software components in a multi-author software design environment to indicate conflicts (see page 259, right column, paragraph 1, page 264, right column, page 265, left column).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe and Atallah to perform the compatibility check in a multi-author environment such that the alert messages are issued to registered authors of the software component as taught by Lin because it is well known in the art that it is desirable to check version compatibility during software development.

31. As per claim 26, Schwab does not teach that the computer has the means to rate each detected difference according to a backward compatibility metric.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including rating each detected difference according to a compatibility metric

and determining an overall compatibility of the software products based on the difference ratings (see Figs 4A, 4B, abstract, lines 2-7 and lines 11-14, column 5, line 64 - column 7, line 15).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwab to rate each detected difference according to a backward compatibility metric as taught by Atallah because it is expensive and time consuming to purchase new versions of application programs and migrate the accompanying data when operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow user to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

32. As per claim 27, Schwab does not teach that the computer has means to incorporate the ratings and the detected differences into the alert message.

Atallah teaches generating a report to indicate the binary compatibility risk profile with records to indicate failure, high risk, low risk, and guaranteed eligible (see Figs 4A, 4B, abstract, lines 10-16, column 5, line 64 - column 7, line 15).

33. As per claim 28, Schwabe teaches the invention as claimed, including a computer-readable medium having stored thereon a plurality of instructions for monitoring updates to a software repository, the plurality of instructions comprising instructions to:

download a first software interface (see Fig 11A, 11C, Fig 20C, column 17, lines 36-67, column 19, lines 19-27, column 24, lines 24-32, column 25, lines 23-36) ;

detect at least one difference between the first software interface and a second software interface (see Figs 20C-20D, column 24, line 35-column 26, line 15); and

issue an alert message containing an overall backward compatibility (see column 25, line 23 – column 26, line 15).

Schwabe does not explicitly teach snapshots of the software interface, and comparing the snapshots. However, a snapshot is considered as a copy of the software interface made at a particular time and applications for taking snapshots of a file are well known in the field, so it would have been obvious to one of ordinary skill in the art that a snapshot of the API definition file can be easily assembled and that comparing two snapshots of two software interfaces would be equivalent to comparing the two software interfaces directly.

Schwabe does not teach rating each detected difference according to a backward compatibility metric, determining an overall backward compatibility of the first software interface with respect to the second software interface, based on the difference ratings.

Atallah teaches a method of assessing the risk of binary compatibility failure between software products, including rating each detected difference according to a compatibility metric and determining an overall compatibility of the software products based on the difference ratings (see Figs 4A, 4B, abstract, lines 2-7, 11-14, column 5, line 64 - column 7, line 15).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe to rate each detected difference according to a backward compatibility metric and determine an overall backward compatibility of the updated version based on the difference ratings as taught by Atallah because it is expensive and time consuming to purchase new versions of application programs and migrate the accompanying data when

operating systems are updated and current versions of application programs are no longer compatible with the new version of the operating system, so the risk profile generated will allow users to better gauge whether upgrading their operating system may create binary compatibility issues with existing application software (see column 1, 42-59, and column 7, lines 5-16 of Atallah).

Schwabe and Atallah do not teach issuing an alert message to registered authors of a software component associated with the software interface in a multi-author software design environment.

Lin teaches issuing alert messages to registered authors of software components in a multi-author software design environment to indicate conflicts (see page 259, right column, paragraph 1, page 264, right column, page 265, left column).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe and Atallah to perform the compatibility check in a multi-author environment such that the alert messages are issued to registered authors of the software component as taught by Lin because it is well known in the art that it is desirable to check version compatibility during software development.

34. As per claim 32, this is a system claim containing limitations that are substantially similar to claim 28. Therefore, it is rejected using the same reasons as claim 28.

35. Claims 29-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Schwabe (US 6,986,132 B1) in view of Atallah et al. (US 7,069,474 B2, hereinafter Atallah), further in view of Lin et al., "Multiuser Collaborative Work in Virtual Environment based CASE Tool" (hereinafter Lin), as applied to claim 28 above, further in view of Theodossy et al. (US 6,898,768 B1, hereinafter Theodossy).

36. As per claim 29, Schwabe, Atallah, and Lin do not teach that the plurality of instructions stored on the machine-readable medium is executed in response to an external trigger.

Theodossy teaches a method for compatibility receiving a compatibility triggering event and verifying compatibility in response to the triggering event (see Fig 5, abstract, lines 4-7, column 3, lines 32-35, and column 6, lines 23-67).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe, Atallah, and Lin such that the instructions are executed in response to an external trigger as taught by Theodossy because compatibility is critical for all upgrades and it is important to verify that compatibility exists for hardware revisions and software revisions prior to experiencing problems relating to incompatibility, and the triggering event allows this verification to occur automatically (see column 2, lines 36-45 and column 3, lines 35-40 of Theodossy).

37. As per claim 30, Schwabe, Atallah, and Lin do not teach that the external trigger comprises a received notification of an update to the second software interface.

Theodossy teaches that the triggering event can be an attempt to perform a software upgrade on a system component (see column Fig 4, column 2, lines 48-55, column 5, line 52 – column 6, line 13, and column 6, line 23-63). While Theodossy does not specifically teach as such, it would have been obvious to one of ordinary skill in the art that the triggering event can be a notification of an update to the second software interface since an triggering event can be any event that necessitates the system to perform a compatibility verification to ensure optimal performance (see column 6, lines 29-33 of Theodossy) and update to a software interface would have an impact on other software interfaces relying on the updated software interface.

38. As per claim 31, Schwabe, Atallah, and Lin do not teach that the external trigger comprises a received notification of a scheduled event.

Theodossy teaches receiving a compatibility triggering event (see column Fig 4, column 2, lines 48-55, column 5, line 52 – column 6, line 13, and column 6, line 23-63). While Theodossy does not explicitly teach that the triggering event is a scheduled event, it would have been obvious to one of ordinary skill in the art that the triggering event could be scheduled since the triggering event can be upgrades on a system component and system upgrades can be a complex process requiring much planning (see column 1, lines 12-13).

39. Claim 34 is rejected under 35 U.S.C. 103(a) as being unpatentable over Schwabe (US 6,986,132 B1) in view of Atallah et al. (US 7,069,474 B2, hereinafter Atallah), further in view of Lin et al., “Multiuser Collaborative Work in Virtual Environment based CASE Tool” (hereinafter Lin), further in view of Boonsiri et al., “Automated Component Ensemble

Evaluation” (hereinafter Boonsiri), as applied to claim 1 above, further in view of Kiessig et al., (US 7,289,973 B2, hereinafter Kiessig).

40. As per claim 34, Schwabe, Atallah, Lin, and Boonsiri do not teach providing an option to store snapshots in a snapshot history.

Kiessig teaches a method of storing snapshots in a snapshot history (see column 5, lines 13-22, 27-36, 44-57).

It would have been obvious to one of ordinary skill in the art at the time of the invention to have modified Schwabe, Atallah, Lin, and Boonsiri to store snapshots in a snapshot history as taught by Kieesig because it allows users to restore past snapshot versions as the current snapshot (see column 5, lines 13-22 of Kiessig).

#### *Response to Arguments*

41. Rejections of claims under §103(a):

42. As per independent claim 1, Applicant’s arguments directed to Boonsiri have been fully considered and are moot in light of the new grounds of rejections.

43. As per independent claim 1, Applicant argued that Atallah does not teach “rating each detected difference according to a backward compatibility metric” because in Atallah’s system, the ratings are each associated with a broad category, and therefore, Atallah does not rate each detected difference. Applicant’s arguments have been fully considered and Examiner respectfully disagrees. Examiner submits that Atallah rates each difference found for the binary differently



because Atallah teaches generating a record for each compatibility test (see Fig 4A, Fig 4B, column 6, line 31 - column 7, line 4). The result of each compatibility test that resulted in a yes answer is considered as one difference (i.e., binary invoked unsupported symbols, binary invokes problem libraries, binary invokes changed libraries, etc.), and each of these differences are recorded separately because a separate record is generated for each difference regardless of answers to the previous compatibility tests as indicated by flow graph on Fig 4A and Fig 4B where flow continues to the next compatibility test after the generation of a record. Atallah further teaches generating an overall compatibility rating based on the difference rating because Atallah teaches “Rather than presenting a simple pass or fail output, the risk assessment system assigns a risk level to the application based on the outcome of the test administered by the appcert application” (column 5, line 67 – column 6, line 3). While Applicant contended that the result of each compatibility test is associated with a broad category, and not each detected difference, Examiner respectfully disagrees with this categorization. Examiner submits that the claim language as currently recited does not specify any specific granularity to be associated with each detected difference. Therefore, a failed result of a compatibility test in Atallah would reasonably be considered as a detected difference and meets the claim language as recited.

44. As per claim 8, Applicant argued that Atallah does not teach “alert messages [which] include[] a summary of each detected difference” because the report of Atallah only contains a result for a few broad categories and does not rate each detected difference. Applicant's arguments have been fully considered and Examiner respectfully disagrees. Examiner submits that the claim language as currently recited does not specify any specific granularity to be

associated with each detected difference. Therefore, a failed result of a compatibility test in Atallah would reasonably be considered as a detected difference and meets the claim language as recited.

### ***Conclusion***

45. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- DeBoskey et al. (US 6,073,161) is cited to teach an apparatus for determining editing conflicts in a multi-authoring system.
- Carter et al. (US 6,519,767 B1) is cited to teach compiler for automatically building version compatible object application.
- Hurley et al. (US 6,678,882 B1) is cited to teach collaborative model for software systems.

46. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jue S. Wang whose telephone number is (571) 270-1655. The examiner can normally be reached on M-Th 7:30 am - 5:00pm (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on 571-272-3759. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Lewis A. Bullock, Jr./  
Supervisory Patent Examiner, Art Unit 2193/

Jue Wang  
Examiner  
Art Unit 2193